2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Differential Evolution with Combined Variants for Dynamic Constrained Optimization

María-Yaneli Ameca-Alducin, Efrén Mezura-Montes and Nicandro Cruz-Ramírez
Facultad de Física e Inteligencia Artificial
Universidad Veracruzana
Sebastian Camacho 5, Col. Centro, CP 91000, Xalapa, Veracruz, Mexico
Email: yaneliameca@gmail.com, {emezura,ncruz}@uv.mx

***Abstract*—In this work a differential evolution algorithm is adapted to solve dynamic constrained optimization problems. The approach is based on a mechanism to detect changes in the objective function and/or the constraints of the problem so as to let the algorithm to promote the diversity in the population while pursuing the new feasible optimum. This is made by combining two popular differential evolution variants and using a memory of best solutions found during the search. Moreover, random-immigrants are added to the population at each generation and a simple hill-climber-based local search operator is applied to promote a faster convergence to the new feasible global optimum. The approach is compared against other recently proposed algorithms in an also recently proposed benchmark. The results show that the proposed algorithm provides a very competitive performance when solving different types of dynamic constrained optimization problems.**

I. Introduction

Evolutionary algorithms (EAs) have been widely used to solve constrained optimization problems [1]–[3]. However, based on a recent review of the state-of-the-art [4], there are some topics which have been scarcely studied. Among them, there is the presence of dynamic constraints [5]. This type of problems are known as Dynamic Constrained Optimization Problems (DCOPs) [5]–[7]. A DCOP can be seen as a single search problem in which a set of constrained optimization problems must be solved during the search process. Given those conditions, traditional EAs must be adapted to identify changes in the search space so as to be able to find new optimal solutions [5].

The specialized literature of EAs shows a significant amount of research in dynamic unconstrained optimization problems [8]. However, in presence of dynamic constraints the research is still scarce. In a recent review on EAs for solving DCOPs [5] it was shown that most of the EAs for such optimization problems are genetic algorithms (GAs). More recently, other nature-inspired meta-heuristics have been adapted to solve DCOPs, as it is the case of the gravitational search algorithm (GSA), whose operation is based on the attraction of particles by gravity [6]. Among the mechanisms added to an EA to solve DCOPs the most popular are the introduction and/or maintenance of diversity [9]–[11] and solution repair [5], [7].

The motivation of this work relies on the interest of testing other EAs in DCOPs and also adding other mechanisms which may help to improve their performance in dynamic constrained search spaces. Differential Evolution (DE) is a very popular EA which has showed a highly competitive performance when solving unconstrained [12] and constrained [13] numerical optimization problems. Moreover, DE has been adapted to solve dynamic unconstrained optimization problems [14]. However, to the best of the authors' knowledge, DE variants have not been combined to solve DCOP's. Therefore, in this paper we propose an adaptation of DE to solve DCOPs by combining two DE variants and temporal modifications to their parameters with the aim to promote exploration after converging to a previous feasible optimum. A simple change detection mechanism is used to activate the variant modification as well as a memory to store the best solutions found during the search. Finally, a simple local search operator is applied at each cycle of the algorithm to favor convergence to the new feasible optimum and some solutions generated at random are inserted in the current population.

The rest of the paper is divided as follows. In Section II the problem of interest is stated. Section III details the DE-based algorithm to solve DCOPs. Section IV presents the experiments and results obtained by the algorithm in a benchmark recently proposed [5]. Finally, Section V includes the conclusions and directions regarding future research.

II. Problem Statement

Without loss of generality, a DCOP can be defined as to: Find $\vec{x}$ which minimizes, at each time t:

$$\min_{\vec{x} \in F_t \subseteq [L,U]} f(\vec{x},t) \tag{1}$$

Subject to:

$$g_i(\vec{x},t) \leq 0, \forall i \in 1,\dots,m \tag{2}$$

$$h_j(\vec{x},t) = 0, \forall j \in 1,\dots,p \tag{3}$$

where $t \in N^+$ is the current time,

$$[L,U] = \{\vec{x} = (x_1, x_2, ..., x_D) | L_i \leq x_i \leq U_i, i = 1 \dots D\} \tag{4}$$

is the search space,

$$F_t = \{\vec{x} | \vec{x} \in [L,U], g_i(\vec{x},t) \leq 0, i = 1 \dots m, \\ h_j(\vec{x},t) = 0, j = 1 \dots p\} \tag{5}$$

is called the feasible region for time t. $\forall \vec{x} \in F_t$ if there exists a solution $\vec{x}^* \in F_t$ such that $f(\vec{x}^*,t) \leq f(\vec{x},t)$, then $\vec{x}^*$ is called a feasible optimal solution and $f(\vec{x}^*,t)$ is called the feasible optima value for time t.

The main features of a DCOP are taken from one of the next four cases: i) a static objective function and static constraints function (i.e. a static constrained optimization problem), ii) a dynamic objective function and static constraints, iii) a static objective function and dynamic constraints, and iv) a dynamic objective function and dynamic constraints.

III. Proposed Approach

A. Differential Evolution

DE [12] is a quite simple but powerful search algorithm which works with a population of solutions called vectors. Two variation operators are applied to the vectors to generate new ones and a greedy selection between parent and offspring is adopted for replacement purposes. The population is represented as shown in Equation 6:

$$\vec{x}_{i,G}, \ i = 1, \ldots, NP \tag{6}$$

where $\vec{x}_{i,G}$ is vector i at generation G, and NP is the number of vectors in the population. Each vector $\vec{x}_{i,G}$ (called target vector) generates one offspring $\vec{u}_{i,G}$ (called trial vector) by using two variation operators as follows: A mutant vector $\vec{v}_{i,G}$ is computed by choosing three vectors ($\vec{x}_{r0,G}$, $\vec{x}_{r1,G}$, and $\vec{x}_{r2,G}$) at random from the current population ($r0 \neq r1 \neq r2 \neq i$) (see Eq. 7).

$$\vec{v}_{i,G} = \vec{x}_{r0,G} + F(\vec{x}_{r1,G} - \vec{x}_{r2,G}) \tag{7}$$

where $F > 0$ is a scale factor defined by the user. After that, the trial (offspring) vector is generated by applying a crossover operator to the target vector $\vec{x}_{i,G}$ and the mutant vector $\vec{v}_{i,G}$ as shown in Eq. 8.

$$u_{i,j,G} = \begin{cases} v_{i,j,G} & \text{if}(rand_i \leq Cr) \text{ or } (j = J_{rand}) \\ x_{i,j,G} & \text{otherwise} \end{cases} \tag{8}$$

where $CR \in [0, 1]$ defines the similarity between the trial vector and the mutant vector, $rand_i$ generates a random number between 0 and 1, and $j \in \{1, \ldots, NP\}$ is the j-th variable of the D- dimensional search space. $j_{rand} \in [1 - D]$ is a random integer number which forces the trial vector to get at least one value inherited from the mutant vector to prevent target vectors cloning.

Finally a greedy selection is made between the target and trial vectors, where the best of them is chosen to remain in the population for the next generation (see Eq 9).

$$\vec{x}_{i,G+1} = \begin{cases} \vec{u}_{i,G} & \text{if}(f(\vec{u}_{i,G}) \leq f(\vec{x}_{i,G})), \\ \vec{x}_{i,G} & \text{otherwise} \end{cases} \tag{9}$$

The complete pseudocode of DE is presented in Algorithm 1. This DE variant is known as DE/rand/1/bin. In this work the DE/best/1/bin is also adopted. The only difference with respect to DE/rand/1/bin is the following: instead of choosing $r0$ at random, this vector is the best one in the current population. Therefore, the expression in Eq. 7 is changed as the one in Eq. 10

$$\vec{v}_{i,G} = \vec{x}_{best,G} + F(\vec{x}_{r1,G} - \vec{x}_{r2,G}) \tag{10}$$

Algorithm 1 Differential Evolution Algorithm (DE/rand/1/bin)

G=0
Create a randomly-generated initial population $\vec{x}_{i,G} \, \forall i, i = 1, \ldots, NP$
Evaluate $f(\vec{x}_{i,G}) \, \forall i, i = 1, \ldots, NP$
for $G \leftarrow 1$ to MAX_GEN **do**
  for $i \leftarrow 1$ to NP **do**
    Randomly select $r0 \neq r1 \neq r2 \neq i$
    $J_{rand} = randint[1, D]$
    for $j \leftarrow 1$ to D **do**
      if $rand_j \leq Cr$ Or $j = J_{rand}$ **then**
        $u_{i,j,G} = x_{r1,j,G} + F(x_{r2,j,G} - x_{r3,j,G})$
      else
        $u_{i,j,G} = x_{i,j,G}$
      end if
    end for
    if $f(\vec{u}_{i,G}) \leq f(\vec{x}_{i,G})$ **then**
      $\vec{x}_{i,G+1} = \vec{u}_{i,G}$
    else
      $\vec{x}_{i,G+1} = \vec{x}_{i,G}$
    end if
  end for
  $G = G + 1$
end for

B. Mechanisms added

In order to deal with a dynamic objective function and/or the dynamic constraints of a DCOP, the change must be detected promptly and the behavior of the search must change as well. After a change in the features of the search space occurs, exploration must be strongly promoted to keep the algorithm from converging to those promising regions of the search space before such change (i.e. the feasible region before the change). Therefore, to an algorithm for DCOPs it must be necessary to leave such a region if a change occurs. On the other hand, as a conflicting goal, a faster convergence to the new feasible global optimum is required before a new change might take place. We defined some terms used in this work: a *cycle* is the period between each environmental change, a *generation* is a complete DE iteration, an *iteration* is a complete cycle of the local search method.

From the above mentioned, a suitable transition between an increased exploration and fast convergence, after an environment change, is required to leave the previous promising feasible region and quickly locate the new one. The constraint-handling technique adopted, the change detection mechanism, and also the exploration promotion mechanism as well as the convergence promotion mechanism are detailed next:

1) Constraint-handling: As it has been reported in the specialized literature [4], the feasibility rules proposed by Deb [15] have been successfully added to DE so as to deal with constrained search spaces. Therefore, in this work such rules are adopted as the constraint-handling technique. They are the following:

1) Between 2 feasible vectors, the one with the highest fitness value is selected.

2) If one vector is feasible and the other one is infeasible, the feasible vector is selected.
3) If both vectors are infeasible, the one with the lowest sum of constraint violation is selected.

Those rules are applied in the greedy selection made in Eq. 9 and also every time the best vector is selected.

2) Change detection: The change detection is an important task for an EA dealing with a DCOP [14], [16]. The first trial vector and the trial vector at the middle of the current population (based on the loop in step 5 in Algorithm 1) are evaluated and their objective function values and constraints values are compared against their previous values. If some of those values are different, a flag value is changed from inactive to active. Finally, the best vector so far in the current population is stored in a memory file. The pseudocode of this mechanism is detailed in Algorithm 2. It is remarked that the change detection mechanism operates twice during a generation because it provides a good trade-off between detecting a change on time and the number of evaluations computed in the process.

Algorithm 2 Change_detection_mechanism

Require: $\vec{x}_{i,t}$
change_detected=FALSE
Evaluate $\vec{x}_{i,t}$ in the objective function and constraints of the DCOP at time t
if at least one value among the objective function and the constraints is not the same as those of the last evaluation of $\vec{x}_{i,t}$ **then**
  change_detected=TRUE
  Copy the best vector in the population $\vec{x}_{best,t}$ to the memory
  Reevaluate all vectors in the current population and also in the memory
  G_using_DE/best/1/bin=0
  NI=IA
end if

3) Exploration promotion: After the detection flag (change_detected) is activated, the DE variant is changed from DE/rand/1/bin (the base variant) to DE/best/1/bin (the alternative variant) for a number of generations defined by the user (G_using_DE/best/1/bin), and the F value is increased to favor larger movements. Furthermore, the best vector can be chosen from either the current population or the memory of best vectors found in previous environments. A previous study on DE for constrained optimization where these two variants were combined [13], showed that DE/best/1/bin, coupled with larger F values, presented a convenient exploration behavior preceding the usage of DE/rand/1/bin in constrained search spaces. The research in this paper is inspired on it and presents its adaptation for DCOPs. The details of this mechanism can be seen in Algorithm 3

Finally, at the end of each generation, a number of randomly generated vectors called immigrants [17] are inserted into the current population. The number of immigrants is modified after a detected change and returns to its original value after DE/best/1/bin finishes its work.

Algorithm 3 Exploration_promotion_mechanism

if the number of G_using_DE/best/1/bin is below its limit **then**
  Generate the trial vector $\vec{u}_{i,t}$ for vector $\vec{x}_{i,t}$ using Eqs 10 and 8 {DE/best/1/bin by choosing the best vector from the memory or the current population}
else
  Generate the trial vector $\vec{u}_{i,t}$ for vector $\vec{x}_{i,t}$ using Eqs 7 and 8 {DE/rand/1/bin}
end if

4) Convergence promotion: A simple hill-climber-like local search operator [18] is applied to a randomly chosen vector $\vec{x}_{rand,t}$ from the current population. A variable chosen at random from $\vec{x}_{rand,t} = (x_{rand,1,t}, x_{rand,2,t}, \dots, x_{rand,D,t})$ is perturbed by a small random value $\delta \in [0, 1]$. Such value is added and subtracted to the variable. The best vector, based on the feasibility rules, among the original one and the two neighbors is chosen as the new starting point. The process is repeated ILS (Iterations for Local Search) times, a parameter defined by the user. The final obtained vector replaces the worst vector in the current population (based on the feasibility rules). The details of the local search operator are in Algorithm 4.

Algorithm 4 Convergence_promotion_mechanism

Select one vector from the current population at random $\vec{x}_{random,t}$
for $c \leftarrow 1$ to ILS **do**
  $\delta = random(0, 1)$
  Select one variable x_i of $\vec{x}_{random,t}$ at random
  Generate neighbor $\vec{x}_{N_i+,t}$ from vector $\vec{x}_{random,t}$ by adding δ to variable x_i
  Generate neighbor $\vec{x}_{N_i-,t}$ from vector $\vec{x}_{random,t}$ by subtracting δ to variable x_i
  From $\vec{x}_{random,t}$, $\vec{x}_{N_i+,t}$, and $\vec{x}_{N_i-,t}$ select the best one based on the feasibility rules and make it the new $\vec{x}_{random,t}$.
end for
Replace the worst vector in the population with the best vector obtained in the previous loop

C. DDECV algorithm for DCOPs

The complete pseudocode of the so-called Dynamic Differential Evolution with Combined Variants (DDECV) to solve DCOPs is presented in Algorithm 5.

IV. Experiments and results

DDECV was tested on a recently proposed benchmark for DCOPs [5], which contains eighteen problems with different features. Due to space restrictions such details are not included in this paper. However, they can be found in [5] and a summary is included in Table I.

To promote a fair comparison, the settings for the benchmark problems adopted in this work are the same as those reported in the reference from where the results for comparison were taken [10], [19] and they are presented in Table II. As it can be noted in such table, the experiment aims to analyze the

TABLE I. MAIN FEATURES OF THE TEST PROBLEMS [5].

Problem	Obj. Function	Constraints	DFR	SwO	bNAO	OICB	OISB	Path
g24_u	Dynamic	No Constraints	1	No	No	No	Yes	N/A
g24_1	Dynamic	Static	2	Yes	No	Yes	No	N/A
g24_f	Static	Static	2	No	No	Yes	No	N/A
g24_uf	Static	No Constraints	1	No	No	No	Yes	N/A
g24_2	Dynamic	Static	2	Yes	No	Yes and No	Yes and No	N/A
g24_2u	Dynamic	No Constraints	1	No	No	No	Yes	N/A
g24_3	Static	Dynamic	2-3	No	Yes	Yes	No	N/A
g24_3b	Dynamic	Dynamic	2-3	Yes	No	Yes	No	N/A
g24_3f	Static	Static	1	No	No	Yes	No	N/A
g24_4	Dynamic	Dynamic	2-3	Yes	No	Yes	No	N/A
g24_5	Dynamic	Dynamic	2-3	Yes	No	Yes and No	Yes and No	N/A
g24_6a	Dynamic	Static	2	Yes	No	No	Yes	Hard
g24_6b	Dynamic	No Constraints	1	No	No	No	Yes	N/A
g24_6c	Dynamic	Static	2	Yes	No	No	Yes	Easy
g24_6d	Dynamic	Static	2	Yes	No	No	Yes	Hard
g24_7	Static	Dynamic	2	No	No	Yes	No	N/A
g24_8a	Dynamic	No Constraints	1	No	No	No	No	N/A
g24_8b	Dynamic	Static	2	Yes	No	Yes	No	N/A

DFR	Number of disconnected feasible regions
SwO	Switched global optimum between disconnected regions
bNAO	Better newly appear optimum without changing existing ones
OICB	Global optimum is in the constraint boundary
OISB	Global optimum is in the search boundary
Path	Indicate if it is easy or difficult to use mutation to travel between feasible regions
Dynamic	The function is dynamic
Static	There is no change
*	In some change periods, the landscape either is a plateau or contains infinite number of optima and all optima (including the existing optimum) lie in a line parallel to one of the axes

TABLE II. PARAMETER VALUES FOR THE TEST PROBLEMS TAKEN FROM [5]

Benchmark problems settings	Number of runs	50
	Number of changes	12
	Frequency change	500, 1000, 2000 Evals.
	Obj. function severity k	0.5
	Constraint severity S	20

TABLE III. DDECV PARAMETER VALUES TUNED WITH THE IRACE TOOL [20]

Pop size	25
Crossover	CR = 0.8399
F before change	F = 0.9644
F after change	FA = 1.0820
Immigrants before change	IB = 5
Immigrates after of change	IA = 3
Number of generations DE/best/1/bin operates after change	16
Iterations for local search	ILS = 8

effect of the change frequency in the algorithms' performance. Three values are considered (500, 1000, and 2000 evaluations).

The parameter values used by DDECV are listed in Table III. Such values were fine-tuned by using the iRace tool [20]. The only parameter which remained fixed was NP=25 vectors.

The performance of DDECV was measured with the offline error proposed in [21]. The offline error is the most popular measure in the specialized literature for DCOPS [5], [6] and is defined as the average of the sum of errors in each cycle divided by the sum of the number of cycles. The offline error is always greater than or equal to zero. This latter value indicates a perfect performance [8]. This measure is defined as indicated in Eq. 11:

$$offline\_error = \frac{1}{n}\sum_{j=1}^{n} e(j) \tag{11}$$

where n is the number of cycles so far and $e(j)$ denotes the best error since the last change gained by the algorithm at cycle j.

$$e(j) = |f(x^*, t) - f(x, t)| \tag{12}$$

where $f(x^*, t)$ denotes the feasible global optima at time t and $f(x, t)$ is the best solution found at generation G.

DDECV was compared against different approaches recently tested when solving DCOPS in [5], [6]: GAnoElit is a GA with no elitism, nonlinear ranking selection, arithmetic crossover and uniform mutation. RIGAnoElit is the same as GAnoElit but with random immigrants. HyperMnoElit is the same as GAnoElit but with hypermutation when solutions are degraded. GAelit is the same as GAnoElit but now with elitism. RIGAelit is the RIGAnoElit version with elitism. HyperMelit is the HyperMnoElit version with elitism. GA+Repair is a GA with a repair mechanism, i.e., if one infeasible solution is generated, it is combined with elements of feasible solutions in the population with the aim to get is feasible before reaching a user-defined limit of attempts. Finally, GSA+Repair is a novel meta-heuristic where a similar repair mechanism as

Algorithm 5 DDECV algorithm

G=0
Create a randomly-generated initial population $\vec{x}_{i,G}\ \forall i, i = 1, \ldots, NP$
Evaluate each $\vec{x}_{i,G}\ \forall i, i = 1, \ldots, NP$
$eval = eval + NP$
while $eval \leq Max\_eval$ **do**
    for $i \leftarrow 1$ to NP **do**
        if $i = 1$ or $i = NP/2$ **then**
            Change_detection_Mechanism ($\vec{x}_{i,G}$) {Algorithm 2}
            $eval = eval + 1$
        end if
        Exploration_promotion_mechanism {Algorithm 3}
        $eval = eval + 1$
        if $f(\vec{u}_{i,G})$ is better than $f(\vec{x}_{i,G})$ based on the feasibility rules **then**
            $\vec{x}_{i,G+1} = \vec{u}_{i,G}$
        else
            $\vec{x}_{i,G+1} = \vec{x}_{i,G}$
        end if
    end for
    if G_using_DE/best/1/bin reached its limit **then**
        NI=IB
    else
        G_using_DE/best/1/bin = G_using_DE/best/1/bin $+1$
        NI=IA
    end if
    Add NI immigrants to the current population and evaluate them
    $eval = eval + NI$
    Convergence_promotion_mechanism {Algorithm 4}
    $eval = eval + 2 * ILS$
    $G = G + 1$
end while

the one used in GA+Repair is considered. DDECV was also compared against versions using just one DE variant (i.e. only DE/rand/1/bin or DE/best/1/bin, both with local search) but the results were poor and are not included due to space restrictions.

A. Results

In Table IV the mean and standard deviation values of the offline error are shown for the six GA-based approaches [7] and also for DDECV in all eighteen test problems with a change frequency of 500 cycles. Table V includes the offline error for the six GA-based approaches and also the values obtained by the GSA+Repair algorithm [6] with a change frequency of 1000 cycles. For GSA+Repair no results were found for other change frequencies. Finally Table VII includes the offline error for the six GA-based approaches with a change frequency of 2000 cycles. In the aforementioned tables the best results are remarked with boldface. The statistical validation was made with the 95%-confidence Wilcoxon test on the results obtained by each algorithm. This validation showed significant differences between DDECV and each one of the seven compared algorithms when the change frequency was 500 evaluations (Table IV). Regarding the results when the change frequency was 1000 (Table V), the differences were significant between DDECV and each compared algorithm with the exception of GSA+REPAIR. Finally, the differences were significant when the change frequency was 2000 evaluations (Table VII) in all cases.

TABLE IV. OFFLINE ERROR VALUES OBTAINED BY DDECV AND THE COMPARED ALGORITHMS WITH A CHANGE FREQUENCY OF 500 EVALUATIONS. DIFFERENCES OBSERVED BETWEEN DDECV AND EACH COMPARED ALGORITHM ARE SIGNIFICANT BASED ON THE 95%-CONFIDENCE WILCOXON TEST.

Algorithms	**Functions**		
	G24-u	**G24-1**	**G24-f**
GAnoElit	0.436(±0.082)	0.807(±0.096)	0.870(±0.094)
RIGAnoElit	0.297(±0.029)	0.649(±0.062)	0.737(±0.075)
HyperMnoElit	0.299(±0.04)	0.532(±0.056)	0.313(±0.102)
GAelit	0.184(±0.034)	0.641(±0.091)	0.175(±0.044)
RIGAelit	0.235(±0.031)	0.496(±0.074)	0.266(±0.081)
HyperMelit	0.163(±0.028)	0.520(±0.081)	0.209(±0.085)
GA+Repair	0.500(±0.064)	0.264(±0.066)	0.077(±0.024)
DDECV	**0.082(±0.011)**	**0.227(±0.067)**	**0.075(±0.019)**
Algorithm	**G24-uf**	**G24-2**	**G24-2u**
GAnoElit	0.675(±0.071)	0.470(±0.087)	0.252(±0.066)
RIGAnoElit	0.501(±0.057)	0.346(±0.048)	0.169(±0.028)
HyperMnoElit	0.179(±0.058)	0.357(±0.055)	0.177(±0.039)
GAelit	0.091(±0.028)	0.372(±0.070)	0.132(±0.034)
RIGAelit	0.125(±0.034)	0.325(±0.043)	0.146(±0.039)
HyperMelit	0.091(±0.020)	0.364(±0.052)	0.115(±0.030)
GA+Repair	0.358(±0.109)	0.298(±0.033)	0.354(±0.043)
DDECV	**0.010(±0.005)**	**0.162(±0.032)**	**0.065(±0.005)**
Algorithm	**G24-3**	**G24-3b**	**G24-3f**
GAnoElit	1.033(±0.147)	0.844(±0.109)	1.385(±0.302)
RIGAnoElit	0.792(±0.101)	0.663(±0.073)	0.899(±0.104)
HyperMnoElit	0.576(±0.087)	0.646(±0.094)	0.343(±0.070)
GAelit	0.375(±0.091)	0.631(±0.116)	0.252(±0.103)
RIGAelit	0.436(±0.056)	0.545(±0.068)	0.264(±0.078)
HyperMelit	0.404(±0.118)	0.557(±0.079)	0.244(±0.122)
GA+Repair	**0.063(±0.012)**	**0.184(±0.035)**	**0.035(±0.012)**
DDECV	0.087(±0.024)	0.225(±0.070)	0.071(±0.025)
Algorithm	**G24-4**	**G24-5**	**G24-6a**
GAnoElit	0.878(±0.160)	0.498(±0.097)	0.750(±0.145)
RIGAnoElit	0.688(±0.062)	0.386(±0.066)	0.523(±0.066)
HyperMnoElit	0.626(±0.077)	0.359(±0.044)	0.517(±0.057)
GAelit	0.646(±0.087)	0.367(±0.059)	1.038(±0.184)
RIGAelit	0.542(±0.079)	0.287(±0.035)	0.534(±0.086)
HyperMelit	0.573(±0.075)	0.324(±0.042)	0.694(±0.098)
GA+Repair	**0.143(±0.035)**	0.196(±0.026)	0.616(±0.106)
DDECV	0.233(±0.081)	**0.195(±0.033)**	**0.267(±0.114)**
Algorithm	**G24-6b**	**G24-6c**	**G24-6d**
GAnoElit	0.588(±0.070)	0.605(±0.096)	0.775(±0.125)
RIGAnoElit	0.429(±0.049)	0.471(±0.036)	0.530(±0.081)
HyperMnoElit	0.452(±0.047)	0.485(±0.063)	0.551(±0.053)
GAelit	0.631(±0.073)	0.666(±0.059)	0.664(±0.126)
RIGAelit	0.436(±0.052)	0.443(±0.034)	0.512(±0.062)
HyperMelit	0.535(±0.053)	0.543(±0.050)	0.584(±0.068)
GA+Repair	0.567(±0.059)	0.518(±0.048)	0.475(±0.075)
DDECV	**0.145(±0.029)**	**0.173(±0.048)**	**0.414(±0.083)**
Algorithm	**G24-7**	**G24-8a**	**G24-8b**
GAnoElit	0.963(±0.129)	0.518(±0.069)	1.006(±0.119)
RIGAnoElit	0.771(±0.065)	0.418(±0.048)	0.882(±0.056)
HyperMnoElit	0.663(±0.083)	0.450(±0.037)	0.805(±0.099)
GAelit	0.441(±0.073)	0.356(±0.04)	0.807(±0.073)
RIGAelit	0.565(±0.073)	0.405(±0.045)	0.758(±0.080)
HyperMelit	0.430(±0.066)	0.355(±0.026)	0.710(±0.09)
GA+Repair	**0.134(±0.026)**	0.341(±0.051)	0.380(±0.074)
DDECV	0.156(±0.038)	**0.292(±0.035)**	**0.332(±0.108)**

The results in Table IV indicate that DDECV outperformed the GA-based approaches in thirteen out of eighteen test problems. The exceptions were g24-3, g24-3b, g24-3f, g24-4 and g24-7, where GA+Repair was better.

The results in Table V indicate that DDECV outperformed the GA-based approaches and the GSA+Repair algorithm in eight out of eighteen test problems. The test problems where DDECV was outperformed by GSA+Repair were g24-u g24-3, g24-3b, g24-3f, g24-4, g24-7, g24-6a, g24-6b, g24-6c, and G24-6d. However, the statistical test indicated that such differences were not significant. Therefore, an additional experiment was made specifically for the frequency change of 1000 evaluations, where the parameters were fine-tuned with the IRACE tool but just considering this frequency. The parameters obtained were: NP=25, CR= 0.9724, F= 0.6133, FA= 1.3083, IB=5, IA= 17, Number of cycles DE/best/1/bin operates after change=16, and ILS= 8. The results are presented in Table VI.

As it can be seen in Table VI, DDECV improved its performance by providing the best results in twelve out of eighteen test problems. The exception were g24-3, g24-3b, g24-3f, g24-4, g24-6d, and g24-7.

Finally, based on the results of Table VII, DDECV outperforms the other six compared algorithms in twelve out of eighteen test problems. GA+Repair was better in the remaining six (g24-1, g24-3, g24-3b, g24-3f, g24-4 and g24-7).

By observing the overall competitive performance showed by DDECV in different test problems with three different change frequencies, it is worth noting that the problems where DDECV is outperformed by other algorithms, regardless the change frequency (g24-3, g24-3b, g24-3f, g24-4, g24-7) have one common feature: the global optimum is in the constraint boundary. This requires further research to provide DDECV with an improved ability to sample the boundaries of the feasible region.

V. Conclusions and Future Work

In this paper, a DE-based algorithm with a combination of DE/rand/1/bin and DE/best/1/bin (DDECV) was proposed to solve DCOPs. The feasibility rules were adopted as the constraint-handling mechanism. Furthermore, three mechanisms were considered to deal with the changes in the constraints and/or the objective function: (1) a change detection mechanism where two vectors are re-evaluated and compared against their previous objective/constraints values. If at least one difference is observed, the best vector in the current population is stored in a memory, (2) an exploration promotion mechanism where DE/best/1/bin is used during some generations and the F value is increased, and (3) a convergence promotion mechanism based on a Hill-climber local search operator. Finally, at each cycle of the algorithm random immigrants are inserted into the current population to help increasing diversity.

DDECV was used to solve eighteen test problems with three different change frequencies and its parameters were fine-tuned by using the iRace tool. DDECV's performance was highly competitive mostly in changes after 500 and 2000 evaluations and it was competitive when changes were after 1000 evaluations. On the other hand, DDECV was outperformed

TABLE V. Offline error values obtained by DDECV and the compared algorithms with a change frequency of 1000 evaluations. Differences observed between DDECV and each compared algorithm are significant based on the 95%-confidence Wilcoxon test. The only exception is remarked in gray.

Algorithms	Functions		
	G24-u	**G24-1**	**G24-f**
GAnoElit	0.298(±0.051)	0.609(±0.064)	0.676(±0.085)
RIGAnoElit	0.221(±0.025)	0.493(±0.045)	0.546(±0.072)
HyperMnoElit	0.206(±0.035)	0.361(±0.065)	0.226(±0.056)
GAelit	0.106(±0.035)	0.459(±0.057)	0.154(±0.083)
RIGAelit	0.149(±0.025)	0.346(±0.046)	0.178(±0.051)
HyperMelit	0.111(±0.026)	0.384(±0.065)	0.149(±0.053)
GA+Repair	0.468(±0.059)	0.226(±0.024)	0.041(±0.011)
GSA+Repair	**0.049(±0.004)**	0.132(±0.015)	0.029(±0.012)
DDECV	0.050(±0.006)	**0.109(±0.033)**	**0.029(±0.010)**
	G24-uf	**G24-2**	**G24-2u**
GAnoElit	0.464(±0.064)	0.356(±0.049)	0.159(±0.041)
RIGAnoElit	0.342(±0.032)	0.264(±0.035)	0.107(±0.019)
HyperMnoElit	0.124(±0.041)	0.257(±0.045)	0.130(±0.022)
GAelit	0.063(±0.022)	0.288(±0.050)	0.073(±0.017)
RIGAelit	0.069(±0.020)	0.246(±0.037)	0.091(±0.024)
HyperMelit	0.053(±0.012)	0.253(±0.043)	0.068(±0.016)
GA+Repair	0.218(±0.018)	0.281(±0.036)	0.294(±0.029)
GSA+Repair	0.047(±0.009)	0.182(±0.019)	0.196(±0.012)
DDECV	**0.004(±0.002)**	**0.126(±0.030)**	**0.054(±0.004)**
	G24-3	**G24-3b**	**G24-3f**
GAnoElit	0.760(±0.099)	0.657(±0.097)	0.886(±0.179)
RIGAnoElit	0.538(±0.047)	0.500(±0.038)	0.651(±0.055)
HyperMnoElit	0.411(±0.052)	0.459(±0.069)	0.256(±0.057)
GAelit	0.289(±0.049)	0.457(±0.084)	0.158(±0.058)
RIGAelit	0.308(±0.048)	0.386(±0.051)	0.167(±0.048)
HyperMelit	0.243(±0.050)	0.394(±0.088)	0.128(±0.051)
GA+Repair	0.156(±0.008)	0.171(±0.019)	0.025(±0.008)
GSA+Repair	**0.028(±0.004)**	**0.076(±0.009)**	**0.009(±0.007)**
DDECV	0.057(±0.018)	0.134(±0.033)	0.032(±0.011)
	G24-4	**G24-5**	**G24-6a**
GAnoElit	0.621(±0.101)	0.379(±0.067)	0.529(±0.108)
RIGAnoElit	0.490(±0.053)	0.293(±0.046)	0.366(±0.030)
HyperMnoElit	0.469(±0.057)	0.275(±0.034)	0.383(±0.051)
GAelit	0.453(±0.075)	0.266(±0.029)	0.674(±0.157)
RIGAelit	0.421(±0.047)	0.240(±0.035)	0.333(±0.050)
HyperMelit	0.426(±0.075)	0.248(±0.039)	0.491(±0.071)
GA+Repair	0.211(±0.015)	0.236(±0.024)	0.431(±0.074)
GSA+Repair	**0.073(±0.012)**	0.153(±0.013)	**0.033(±0.003)**
DDECV	0.131(±0.032)	**0.126(±0.019)**	0.215(±0.067)
	G24-6b	**G24-6c**	**G24-6d**
GAnoElit	0.448(±0.054)	0.446(±0.041)	0.543(±0.127)
RIGAnoElit	0.331(±0.035)	0.329(±0.039)	0.366(±0.040)
HyperMnoElit	0.340(±0.046)	0.323(±0.037)	0.370(±0.046)
GAelit	0.408(±0.057)	0.441(±0.052)	0.510(±0.075)
RIGAelit	0.309(±0.039)	0.325(±0.029)	0.342(±0.057)
HyperMelit	0.390(±0.039)	0.394(±0.051)	0.456(±0.041)
GA+Repair	0.427(±0.048)	0.390(±0.038)	0.354(±0.038)
GSA+Repair	**0.047(±0.003)**	**0.045(±0.004)**	**0.037(±0.007)**
DDECV	0.108(±0.016)	0.128(±0.025)	0.288(±0.055)
	G24-7	**G24-8a**	**G24-8b**
GAnoElit	0.721(±0.088)	0.426(±0.050)	0.835(±0.068)
RIGAnoElit	0.543(±0.059)	0.346(±0.031)	0.719(±0.071)
HyperMnoElit	0.495(±0.053)	0.374(±0.043)	0.681(±0.072)
GAelit	0.316(±0.053)	0.266(±0.028)	0.662(±0.056)
RIGAelit	0.416(±0.068)	0.304(±0.028)	0.598(±0.064)
HyperMelit	0.315(±0.062)	0.279(±0.028)	0.608(±0.071)
GA+Repair	0.181(±0.017)	0.496(±0.032)	0.391(±0.068)
GSA+Repair	**0.018(±0.002)**	0.202(±0.041)	0.192(±0.034)
DDECV	0.106(±0.022)	**0.141(±0.025)**	**0.151(±0.058)**

TABLE VI. OFFLINE ERROR VALUES OBTAINED BY DDECV AND THE COMPARED ALGORITHMS WITH A CHANGE FREQUENCY OF 1000 EVALUATIONS WITH FINE-TUNED PARAMETERS FOR THIS FREQUENCY. DIFFERENCES OBSERVED BETWEEN DDECV AND EACH COMPARED ALGORITHM ARE SIGNIFICANT BASED ON THE 95%-CONFIDENCE WILCOXON TEST. THE ONLY EXCEPTION IS REMARKED IN GRAY.

Algorithms	**Functions**		
	G24-u	**G24-1**	**G24-f**
GAnoElit	0.298(±0.051)	0.609(±0.064)	0.676(±0.085)
RIGAnoElit	0.221(±0.025)	0.493(±0.045)	0.546(±0.072)
HyperMnoElit	0.206(±0.035)	0.361(±0.065)	0.226(±0.056)
GAelit	0.106(±0.035)	0.459(±0.057)	0.154(±0.083)
RIGAelit	0.149(±0.025)	0.346(±0.046)	0.178(±0.051)
HyperMelit	0.111(±0.026)	0.384(±0.065)	0.149(±0.053)
GA+Repair	0.468(±0.059)	0.226(±0.024)	0.041(±0.011)
GSA+Repair	0.049(±0.004)	0.132(±0.015)	0.029(±0.012)
DDECV	**0.017(±0.002)**	**0.080(±0.020)**	**0.029(±0.011)**
	G24-uf	**G24-2**	**G24-2u**
GAnoElit	0.464(±0.064)	0.356(±0.049)	0.159(±0.041)
RIGAnoElit	0.342(±0.032)	0.264(±0.035)	0.107(±0.019)
HyperMnoElit	0.124(±0.041)	0.257(±0.045)	0.130(±0.022)
GAelit	0.063(±0.022)	0.288(±0.050)	0.073(±0.017)
RIGAelit	0.069(±0.020)	0.246(±0.037)	0.091(±0.024)
HyperMelit	0.053(±0.012)	0.253(±0.043)	0.068(±0.016)
GA+Repair	0.218(±0.018)	0.281(±0.036)	0.294(±0.029)
GSA+Repair	0.047(±0.009)	0.182(±0.019)	0.196(±0.012)
DDECV	**0.005(±0.003)**	**0.047(±0.018)**	**0.002(±0.003)**
	G24-3	**G24-3b**	**G24-3f**
GAnoElit	0.760(±0.099)	0.657(±0.097)	0.886(±0.179)
RIGAnoElit	0.538(±0.047)	0.500(±0.038)	0.651(±0.055)
HyperMnoElit	0.411(±0.052)	0.459(±0.069)	0.256(±0.057)
GAelit	0.289(±0.049)	0.457(±0.084)	0.158(±0.058)
RIGAelit	0.308(±0.048)	0.386(±0.051)	0.167(±0.048)
HyperMelit	0.243(±0.050)	0.394(±0.088)	0.128(±0.051)
GA+Repair	0.156(±0.008)	0.171(±0.019)	0.025(±0.008)
GSA+Repair	**0.028(±0.004)**	**0.076(±0.009)**	**0.009(±0.007)**
DDECV	0.087(±0.018)	0.116(±0.030)	0.029(±0.010)
	G24-4	**G24-5**	**G24-6a**
GAnoElit	0.621(±0.101)	0.379(±0.067)	0.529(±0.108)
RIGAnoElit	0.490(±0.053)	0.293(±0.046)	0.366(±0.030)
HyperMnoElit	0.469(±0.057)	0.275(±0.034)	0.383(±0.051)
GAelit	0.453(±0.075)	0.266(±0.029)	0.674(±0.157)
RIGAelit	0.421(±0.047)	0.240(±0.035)	0.333(±0.050)
HyperMelit	0.426(±0.075)	0.248(±0.039)	0.491(±0.071)
GA+Repair	0.211(±0.015)	0.236(±0.024)	0.431(±0.074)
GSA+Repair	**0.073(±0.012)**	0.153(±0.013)	0.033(±0.003)
DDECV	0.112(±0.032)	**0.068(±0.017)**	**0.023(±0.012)**
	G24-6b	**G24-6c**	**G24-6d**
GAnoElit	0.448(±0.054)	0.446(±0.041)	0.543(±0.127)
RIGAnoElit	0.331(±0.035)	0.329(±0.039)	0.366(±0.040)
HyperMnoElit	0.340(±0.046)	0.323(±0.037)	0.370(±0.046)
GAelit	0.408(±0.057)	0.441(±0.052)	0.510(±0.075)
RIGAelit	0.309(±0.039)	0.325(±0.029)	0.342(±0.057)
HyperMelit	0.390(±0.039)	0.394(±0.051)	0.456(±0.041)
GA+Repair	0.427(±0.048)	0.390(±0.038)	0.354(±0.038)
GSA+Repair	0.047(±0.003)	0.045(±0.004)	**0.037(±0.007)**
DDECV	**0.017(±0.004)**	**0.022(±0.008)**	0.071(±0.014)
	G24-7	**G24-8a**	**G24-8b**
GAnoElit	0.721(±0.088)	0.426(±0.050)	0.835(±0.068)
RIGAnoElit	0.543(±0.059)	0.346(±0.031)	0.719(±0.071)
HyperMnoElit	0.495(±0.053)	0.374(±0.043)	0.681(±0.072)
GAelit	0.316(±0.053)	0.266(±0.028)	0.662(±0.056)
RIGAelit	0.416(±0.068)	0.304(±0.028)	0.598(±0.064)
HyperMelit	0.315(±0.062)	0.279(±0.028)	0.608(±0.071)
GA+Repair	0.181(±0.017)	0.496(±0.032)	0.391(±0.068)
GSA+Repair	**0.018(±0.002)**	0.202(±0.041)	0.192(±0.034)
DDECV	0.060(±0.017)	**0.042(±0.007)**	**0.042(±0.012)**

TABLE VII. OFFLINE ERROR VALUES OBTAINED BY DDECV AND THE COMPARED ALGORITHMS WITH A CHANGE FREQUENCY OF 2000 EVALUATIONS. DIFFERENCES OBSERVED BETWEEN DDECV AND EACH COMPARED ALGORITHM ARE SIGNIFICANT BASED ON THE 95%-CONFIDENCE WILCOXON TEST.

Algorithms	**Functions**		
	G24-u	**G24-1**	**G24-f**
GAnoElit	0.230(±0.017)	0.464(±0.064)	0.551(±0.050)
RIGAnoElit	0.144(±0.021)	0.356(±0.037)	0.393(±0.054)
HyperMnoElit	0.156(±0.018)	0.278(±0.049)	0.149(±0.050)
GAelit	0.065(±0.011)	0.332(±0.074)	0.092(±0.052)
RIGAelit	0.110(±0.014)	0.235(±0.038)	0.106(±0.037)
HyperMelit	0.072(±0.015)	0.289(±0.053)	0.084(±0.042)
GA+Repair	0.262(±0.040)	**0.055(±0.012)**	0.023(±0.006)
DDECV	**0.030(±0.008)**	0.066(±0.018)	**0.016(±0.004)**
	G24-uf	**G24-2**	**G24-2u**
GAnoElit	0.373(±0.046)	0.264(±0.043)	0.133(±0.014)
RIGAnoElit	0.263(±0.022)	0.200(±0.028)	0.073(±0.010)
HyperMnoElit	0.099(±0.040)	0.192(±0.019)	0.091(±0.021)
GAelit	0.032(±0.010)	0.183(±0.024)	0.049(±0.008)
RIGAelit	0.047(±0.015)	0.168(±0.023)	0.057(±0.011)
HyperMelit	0.028(±0.008)	0.172(±0.037)	0.044(±0.012)
GA+Repair	0.164(±0.054)	0.147(±0.022)	0.171(±0.040)
DDECV	**0.002(±0.001)**	**0.071(±0.016)**	**0.031(±0.002)**
	G24-3	**G24-3b**	**G24-3f**
GAnoElit	0.508(±0.065)	0.460(±0.042)	0.594(±0.073)
RIGAnoElit	0.418(±0.031)	0.350(±0.048)	0.425(±0.049)
HyperMnoElit	0.313(±0.045)	0.343(±0.036)	0.141(±0.056)
GAelit	0.164(±0.033)	0.320(±0.058)	0.072(±0.032)
RIGAelit	0.208(±0.026)	0.262(±0.024)	0.100(±0.026)
HyperMelit	0.168(±0.029)	0.288(±0.048)	0.082(±0.036)
GA+Repair	**0.019(±0.004)**	**0.044(±0.009)**	**0.010(±0.003)**
DDECV	0.032(±0.008)	0.078(±0.015)	0.017(±0.006)
	G24-4	**G24-5**	**G24-6a**
GAnoElit	0.466(±0.079)	0.284(±0.043)	0.375(±0.058)
RIGAnoElit	0.364(±0.034)	0.214(±0.024)	0.250(±0.029)
HyperMnoElit	0.357(±0.045)	0.206(±0.029)	0.257(±0.028)
GAelit	0.333(±0.074)	0.196(±0.026)	0.408(±0.050)
RIGAelit	0.309(±0.037)	0.174(±0.022)	0.236(±0.026)
HyperMelit	0.287(±0.067)	0.182(±0.019)	0.287(±0.036)
GA+Repair	**0.044(±0.009)**	0.111(±0.023)	0.300(±0.054)
DDECV	0.073(±0.014)	**0.081(±0.011)**	**0.103(±0.032)**
	G24-6b	**G24-6c**	**G24-6d**
GAnoElit	0.309(±0.035)	0.318(±0.038)	0.380(±0.076)
RIGAnoElit	0.229(±0.022)	0.243(±0.029)	0.274(±0.033)
HyperMnoElit	0.231(±0.025)	0.237(±0.024)	0.272(±0.027)
GAelit	0.274(±0.028)	0.282(±0.033)	0.318(±0.059)
RIGAelit	0.210(±0.025)	0.213(±0.027)	0.242(±0.027)
HyperMelit	0.234(±0.019)	0.249(±0.034)	0.281(±0.030)
GA+Repair	0.306(±0.030)	0.287(±0.042)	0.263(±0.024)
DDECV	**0.053(±0.008)**	**0.063(±0.013)**	**0.139(±0.027)**
	G24-7	**G24-8a**	**G24-8b**
GAnoElit	0.530(±0.057)	0.371(±0.032)	0.711(±0.052)
RIGAnoElit	0.398(±0.040)	0.315(±0.023)	0.566(±0.067)
HyperMnoElit	0.369(±0.033)	0.310(±0.025)	0.553(±0.063)
GAelit	0.217(±0.047)	0.232(±0.023)	0.499(±0.048)
RIGAelit	0.303(±0.043)	0.269(±0.017)	0.496(±0.042)
HyperMelit	0.253(±0.036)	0.237(±0.013)	0.463(±0.052)
GA+Repair	**0.050(±0.015)**	0.247(±0.020)	0.136(±0.035)
DDECV	0.062(±0.014)	**0.072(±0.014)**	**0.078(±0.032)**

mainly by GA+Repair and GSA+Repair in problems where the feasible global optimum was located at the boundaries of the feasible region. Nevertheless, DDECV was able to improve its results in test problems with a change frequency of 1000 evaluations by increasing the CR, F and the number of immigrants after the change.

Part of the future work includes the addition of a repair mechanism to help DDECV to improve the results in problems where the optimum is located in the boundaries of the feasible region. Moreover, other measures will be calculated [5], [8] and the test problems will be solved by varying the severity of the changes in the values of the objective function and constraints. Finally, DDECV will be tested on real-world applications.

ACKNOWLEDGMENT

The authors acknowledge the valuable help and comments provided by Prof. Swagatam Das.

REFERENCES

[1] Z. Michalewicz and M. Schoenauer, "Evolutionary Algorithms for Constrained Parameter Optimization Problems," *Evolutionary Computation*, vol. 4, no. 1, pp. 1–32, 1996.

[2] C. A. Coello Coello, "Theoretical and Numerical Constraint Handling Techniques used with Evolutionary Algorithms: A Survey of the State of the Art," *Computer Methods in Applied Mechanics and Engineering*, vol. 191, no. 11-12, pp. 1245–1287, January 2002.

[3] E. Mezura-Montes, Ed., *Constraint-Handling in Evolutionary Optimization*, ser. Studies in Computational Intelligence. Springer-Verlag, 2009, vol. 198.

[4] E. Mezura-Montes and C. A. C. Coello, "Constraint-handling in nature-inspired numerical optimization: Past, present and future," *Swarm and Evolutionary Computation*, vol. 1, no. 4, pp. 173–194, 2011.

[5] T. Nguyen and X. Yao, "Continuous dynamic constrained optimization: The challenges," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 6, pp. 769–786, 2012.

[6] K. Pal, C. Saha, S. Das, and C. Coello-Coello, "Dynamic constrained optimization with offspring repair based gravitational search algorithm," in *Evolutionary Computation (CEC), 2013 IEEE Congress on*, 2013, pp. 2414–2421.

[7] T. T. Nguyen and X. Yao, "Benchmarking and solving dynamic constrained problems," in *Evolutionary Computation, 2009. CEC '09. IEEE Congress on*, 2009, pp. 690–697.

[8] T. Nguyen, S. Yang, and J. Branke, "Evolutionary dynamic optimization: A survey of the state of the art," *Swarm and Evolutionary Computation*, vol. 6, no. 0, pp. 1 – 24, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2210650212000363

[9] H. Cobb, "An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments," Naval Research Lab Washington DC, Tech. Rep., 1990.

[10] H. Cobb and J. Grefenstette, "Genetic algorithms for tracking changing environments." in *ICGA*, S. Forrest, Ed. Morgan Kaufmann, 1993, pp. 523–530.

[11] H. Singh, A. Isaacs, T. Nguyen, T. Ray, and X. Yao, "Performance of infeasibility driven evolutionary algorithm (idea) on constrained dynamic single objective optimization problems," in *Evolutionary Computation, 2009. CEC '09. IEEE Congress on*, 2009, pp. 3127–3134.

[12] K. Price, R. Storn, and J. Lampinen, *Differential Evolution A Practical Approach to Global Optimization*, ser. Natural Computing Series. Springer-Verlag, 2005. [Online]. Available: http://www.springer.com/west/home/computer/foundations?SGWID=4-156-22-32104365-0\&\#38;teaserId=68063\&\#38;CENTER_ID=69103

[13] E. Mezura-Montes, M. E. Miranda-Varela, and R. del Carmen Gómez-Ramón, "Differential evolution in constrained numerical optimization. an empirical study," *Information Sciences*, vol. 180, no. 22, pp. 4223–4262, 2010.

[14] M. du Plessis, "Adaptive multi-population differential evolution for dynamic environments," Ph.D. dissertation, Faculty of Engineering, Built Environment and Information Technology, University of Pretoria, April 2012.

[15] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, no. 24, pp. 311–338, 2000.

[16] H. Richter, "Detecting change in dynamic fitness landscapes," in *Evolutionary Computation, 2009. CEC '09. IEEE Congress on*, 2009, pp. 1613–1620.

[17] Y. Shengxiang, "Memory-based immigrants for genetic algorithms in dynamic environments," in *Proceedings of the 2005 conference on Genetic and evolutionary computation*, ser. GECCO '05. New York, NY, USA: ACM, 2005, pp. 1115–1122. [Online]. Available: http://doi.acm.org/10.1145/1068009.1068196

[18] S. Hernandez, G. Leguizamon, and E. Mezura-Montes, "A hybrid version of differential evolution with two differential mutation operators applied by stages," in *Evolutionary Computation (CEC), 2013 IEEE Congress on*, 2013, pp. 2895–2901.

[19] T. Nguyen and X. Yao, "Detailed experimental results of ga, riga, hyperm and ga+repair on the g24 set of benchmark problems," School Comput. Sci., Univ. Birmingham, Birmingham, U.K., Tech. Rep., 2010, disponible en: http://www.staff.livjm.ac.uk/enrtngu1/ Papers/DCOP full-data.pdf.

[20] M. López-Ibáñez and T. Stützle, "Automatically improving the anytime behaviour of optimisation algorithms," IRIDIA, Université Libre de Bruxelles, Belgium, Tech. Rep. TR/IRIDIA/2012-012, May 2012, published in European Journal of Operations Research [22].

[21] J. Branke and H. Schmeck, "Designing evolutionary algorithms for dynamic optimization problems," in *Advances in Evolutionary Computing*, ser. Natural Computing Series, A. Ghosh and S. Tsutsui, Eds. Springer Berlin Heidelberg, 2003, pp. 239–262. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-18965-4_9

[22] A. Radulescu, M. López-Ibáñez, and T. Stützle, "Automatically improving the anytime behaviour of multiobjective evolutionary algorithms," in *Evolutionary Multi-Criterion Optimization*, ser. Lecture Notes in Computer Science, R. Purshouse, P. Fleming, C. M. Fonseca, S. Greco, and J. Shaw, Eds. Springer Berlin Heidelberg, 2013, vol. 7811, pp. 825–840.